# COMPUTER SYSTEMS ORGANIZATION
# CHAPTER 5

1

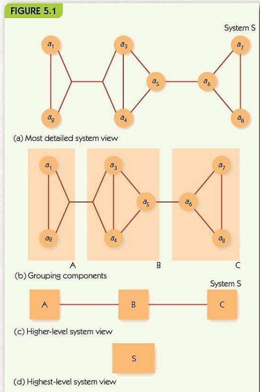
## INTRODUCTION (1 OF 2)

- This chapter changes the **level of abstraction** and focuses on a higher level of computer system construction
- Focus on **functional units** and **computer organization**
- A **hierarchy of abstractions** hides unnecessary details
- Change focus from transistors to gates and to circuits as the basic unit
- Discusses in detail the Von Neumann architecture

2

## INTRODUCTION (2 OF 2)

FIGURE 5.1




The concept of abstraction

3

## THE COMPONENTS OF A COMPUTER SYSTEM

**Von Neumann architecture** is the foundation for nearly all modern computers - https://www.youtube.com/watch?v=Ml3-kVYLNr8 anecdotes about Von Neumann and his times

Three characteristics of the Von Neumann Architecture:

- Four major subsystems of the Von Neumann architecture
  1. Memory
  2. Input/output
  3. Arithmetic/logic unit (ALU)
  4. Control Unit

     ALU and control unit are often bundled inside the **central processing unit (CPU)**
- The stored program concept
- Sequential execution of instructions

4

## VON NEUMANN ARCHITECTURE

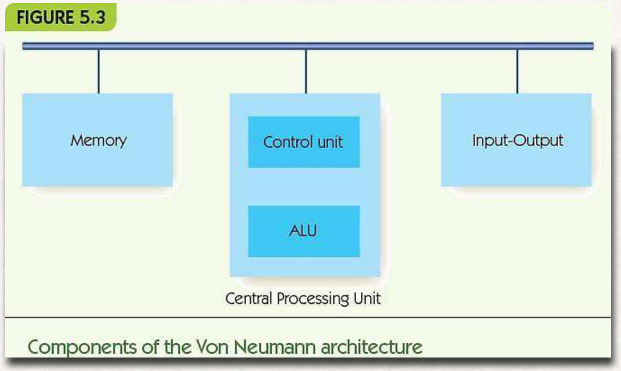



FIGURE 5.3

Components of the Von Neumann architecture

5

## THE ARITHMETIC/LOGIC UNIT

- ALU is part of the **processor**
- Contains **circuits** for arithmetic
  - Addition, subtraction, multiplication, and division
- Contains **circuits** for comparison and logic
  - Equality, and, or, not
- Contains **registers**: high-speed, dedicated memory connected to circuits
- **Data path:** how information flows in the ALU
  - From registers to circuits
  - From circuits back to registers

6

## A SIMPLE 3 REGISTER ALU

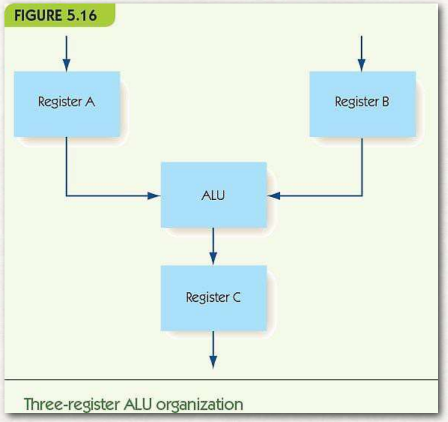



FIGURE 5.16

Three-register ALU organization

7

## A 16 REGISTER ALU

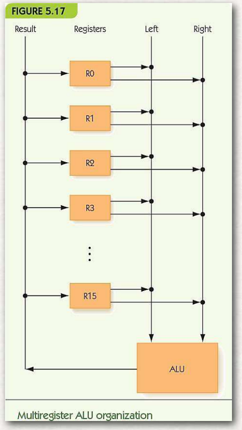



FIGURE 5.17

Multiregister ALU organization

What advantage do we get by increasing the number of registers?

What if we want to do: $(9 + 2) \times (4 - 3) / 6$

8

## THE COMPONENTS OF A COMPUTER SYSTEM THE ALU (1 OF 4)

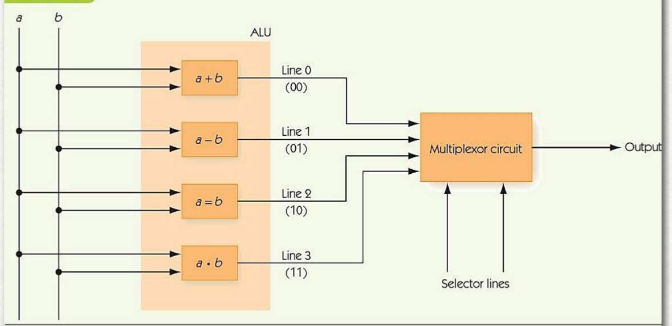
- How is the operation to perform chosen?
  - run all circuits, multiplexor selects one output from all circuits.
- e.g. an ALU which can do +, -, =, AND (these could be represented with bits 00, 01, 10, 11) these bits could be the selector bits for the multiplexor

9

9

## THE COMPONENTS OF A COMPUTER SYSTEM THE ALU (2 OF 4)

FIGURE 5.18




Using a multiplexor circuit to select the proper ALU result

10

## THE COMPONENTS OF A COMPUTER SYSTEM THE ALU (3 OF 4)

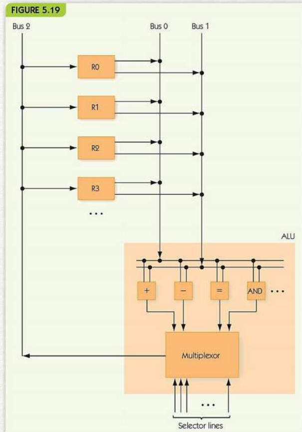
Information flow

- Data comes in from outside to registers
- Signal comes from registers to ALU
- Signal moves from ALU to multiplexor
- Multiplexor selects the value to keep and discards the rest
- Result from the multiplexor goes back to the register and then to outside

11

## THE COMPONENTS OF A COMPUTER SYSTEM THE ALU (4 OF 4)

FIGURE 5.19

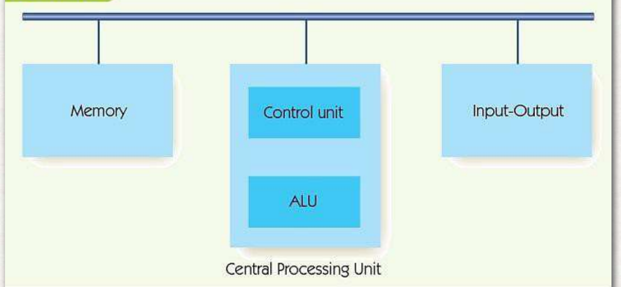



Overall ALU organization

12

## VON NEUMANN ARCHITECTURE (CONTROL UNIT)

FIGURE 5.3




Components of the Von Neumann architecture





## THE CONTROL UNIT

- As we move towards a computer we need to encode what we want the computer to do into values (numbers)
- These will eventually be called opcodes or operation codes. i.e. they represent the task or the instruction we want the computer to perform
- If we have 4 operations (e.g. +, -, =, AND) we can encode the operation with 2 bits
- We also need to encode the operands (what we are adding etc)
- If we want to add any two registers we must be able to identify them (e.g. if 16 registers we need 4 bits for each register)
- We may use memory addresses as the source of our operands as well





## CONTROL UNIT AND PROGRAMS

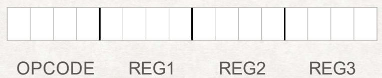
- **Stored program** characteristic
  - Programs are encoded in binary and stored in computer's memory
- **Control unit** fetches instructions from memory, decodes them, and executes them
  - Fetch - get the instruction from memory - the instruction is just a number of bits
  - Decode - using the bits in the instruction the control unit works out what has to be done and on what
    - this is another place a decoder can be used
  - Execute - the control unit is responsible for controlling what happens to make sure that the correct operands are used and then carry out the instruction





## EXAMPLE

- Imagine an instruction which consists of 16 bits
  - If we have 16 different instructions
    - 4 bits could be used to represent the instruction
  - That leaves 12 bits to represent the operands
    - If we have 16 registers we can use we need 4 bits to represent each register.
  - An instruction could look like:



## STATE MACHINE

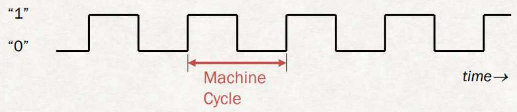
- Controlling even simple computers in this way is complicated
- We can think of the control unit as a state machine
- A state machine …
  - … at each point in time has a state (i.e. registers and other latches have a particular value)
  - … changes from one state to another as input arrives (or in the computer's case there is a clock which alternates between zero and one). We say this clock drives the computation.
- A single instruction can take several clock cycles (and hence move through several states of the state machine)

17

17

## THE CLOCK

- The hardware clock (not the same as a "time on the wall" clock) provides a simple alternation of 0s and 1s like this:
- Each time the clock ticks the CPU can process one instruction.

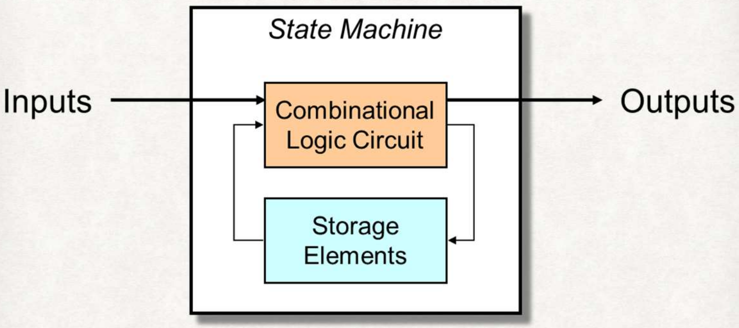



| Cycles per second | Clock rate / clock speed |
|---|---|
| 1 | 1 Hz |
| 1 million | 1 MHZ |
| 3 billion | 3 GHz |

- The line coming from the clock alternates between 0 and 1.
  - When 0 the other circuits settle to a stable value.
  - When 1 the other circuits can change state to a new value.

18

## STATE MACHINE



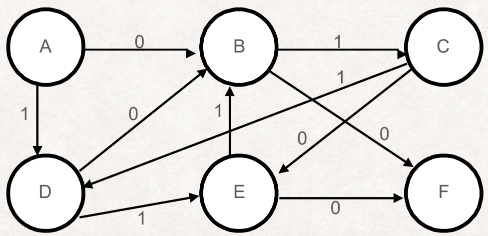

What changes each clock cycle is the output and what is stored in the storage elements. The state machine is thus a sequential circuit.

19

19

## STATE DIAGRAMS

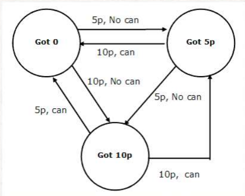
- We can represent state machines in state diagrams. We show the states and the values which cause the transitions from one state to the other
- If we start in state A where do we end up after input of "01101"?
  - What about "010101"?




20

20

## STATE MACHINE EXAMPLES

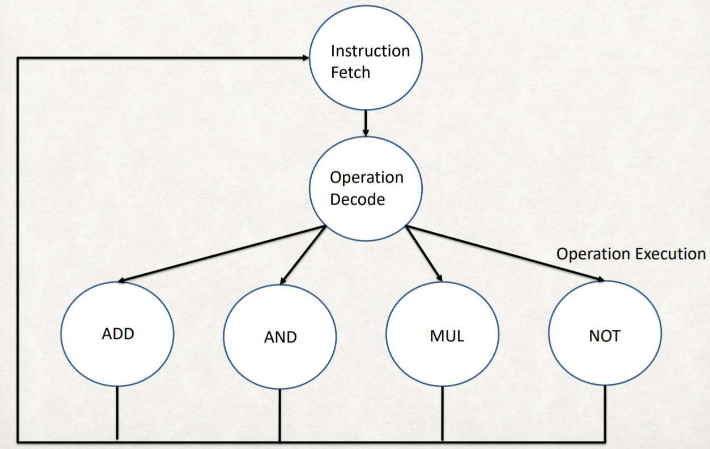



**Vending Machine example**

- The machine accepts only 5p and 10p coinage. When 15p has been inserted into the machine, a can of carbonated sugar water is released
- Later on, when you are looking at Turing machines as a way to represent computation you will see one of the most important parts of a Turing machine is the state of the machine, this is used to work out what to do next in association with the current input.

21

## STATE MACHINE EXAMPLES




22